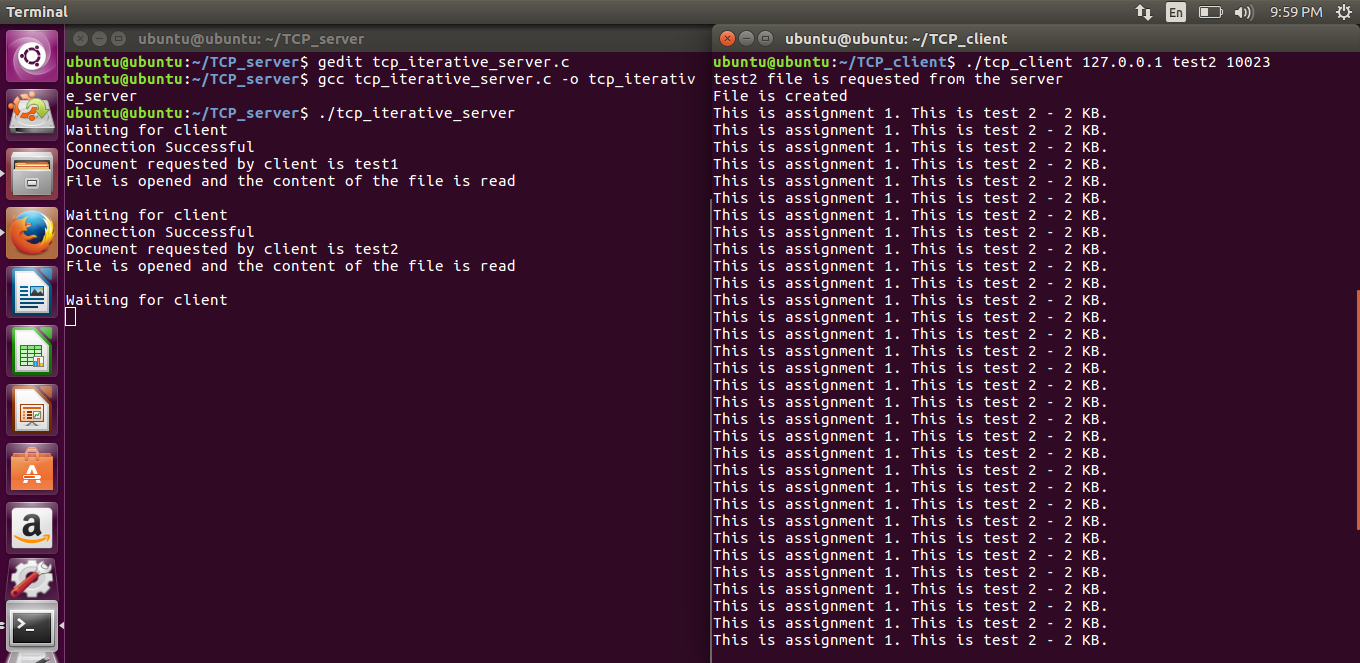
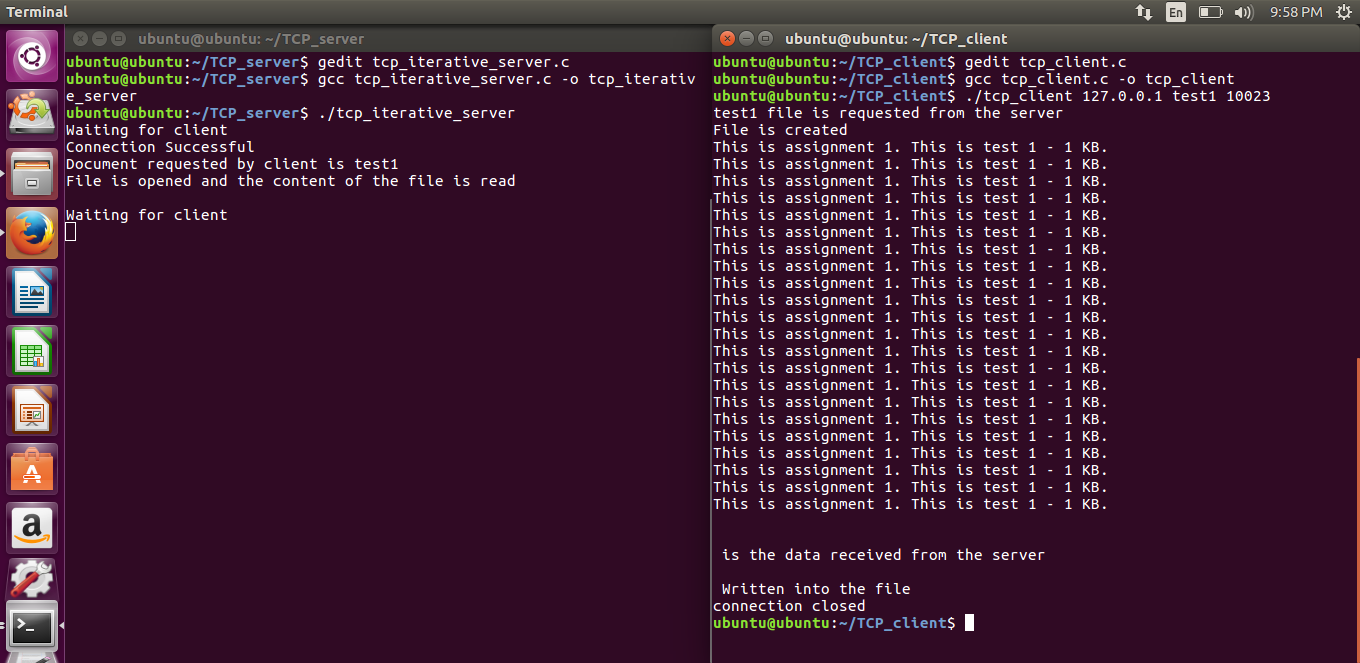
**Lab Assignment - 2 output**

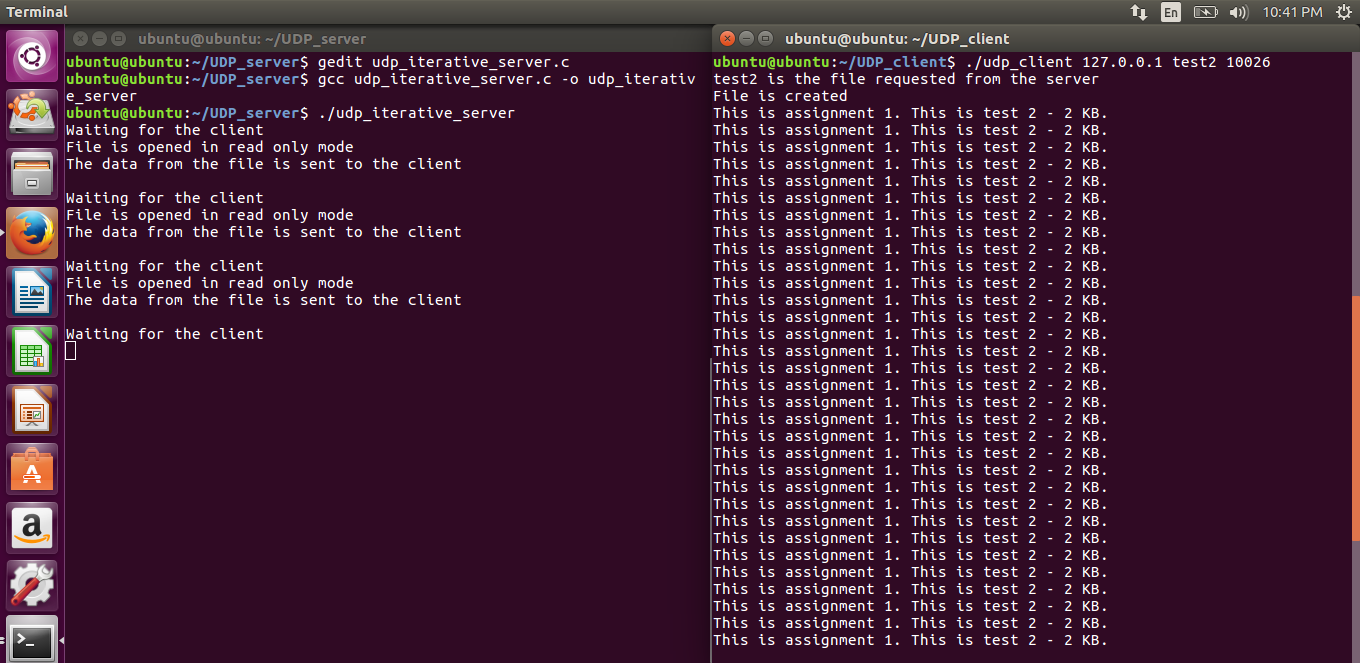
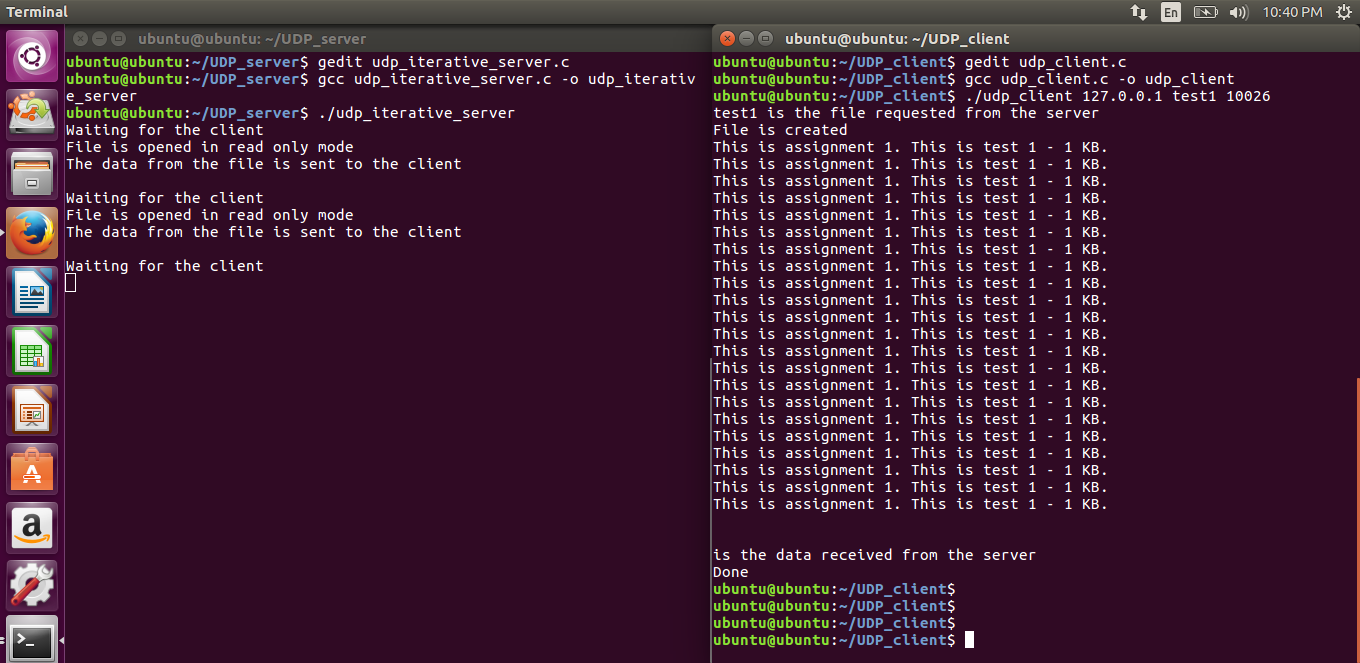
**Gnanadeep Kanthi Usha**

1. **Iterative Server and Client**

**TCP:**



**UDP:**



**2. Concurrent Multiprocessing TCP File server and TCP client with one process per request (On demand child process creation)**

**Server Output:**

kanti@kanti-linux:~/kanti\_sem\_3/207/Assgn\_2/TCP\_fork\_server\_client$ ./server 10000  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Concurrent TCP File Server Program: listening on socket 10000..  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Child processes forked on demand.One per client connection:   
  
New connection request from 127.0.0.1:39328 ==>   
forking new child process 31338 to handle the client..   
File request from client: two\_k\_file.txt  
  
File transfer done.  
  
31338 ==> Closing client connection. Child exiting..  
  
New connection request from 127.0.0.1:39330 ==>   
forking new child process 31355 to handle the client..   
File request from client: one\_k\_file.txt  
  
File transfer done.  
  
31355 ==> Closing client connection. Child exiting..  
  
New connection request from 127.0.0.1:39332 ==>   
forking new child process 31362 to handle the client..   
File request from client: one\_k\_frle.txt  
  
File does not exist. Error msg sent back...  
Closing client connection. child exiting...

**Client Output:**

kanti@kanti-linux:~/kanti\_sem\_3/207/Assgn\_2/TCP\_fork\_server\_client$ ./client 127.0.0.1 10000 one\_k\_file.txt   
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
TCP test client program. Requesting one\_k\_file.txt file from server.  
File will be stored in the client local current directory as "test\_file.txt"  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Connection successful.Requesting file transfer..  
File transfer successful. Closing the socket.Exiting the child.  
kanti@kanti-linux:~/kanti\_sem\_3/207/Assgn\_2/TCP\_fork\_server\_client$ cat test\_file.txt   
start test\_file\_1  
Given a pathname for a file, open() returns a file descriptor, a small, non-negative integer for use in subsequent system calls (read(2), write(2), lseek(2), fcntl(2), etc.). The file descriptor returned by a successful call will be the lowest-numbered file descriptor not currently open for the process.  
The new file descriptor is set to remain open across an execve(2) (i.e., the FD\_CLOEXEC file descriptor flag described in fcntl(2) is initially disabled). The file offset is set to the beginning of the file (see lseek(2)).  
  
A call to open() creates a new open file description, an entry in the system-wide table of open files. This entry records the file offset and the file status flags (modifiable via the fcntl() F\_SETFL operation). A file descriptor is a reference to one of these entries; this reference is unaffected if pathname is subsequently removed or modified to refer to a different file. The new open file description is initially not shared with any other process, but sharing may arise via fork(2).  
end test\_file\_1  
  
kanti@kanti-linux:~/kanti\_sem\_3/207/Assgn\_2/TCP\_fork\_server\_client$ ./client 127.0.0.1 10000 one\_k\_frle.txt   
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
TCP test client program. Requesting one\_k\_frle.txt file from server.  
File will be stored in the client local current directory as "test\_file.txt"  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Connection successful.Requesting file transfer..  
File transfer successful. Closing the socket.Exiting the child.  
kanti@kanti-linux:~/kanti\_sem\_3/207/Assgn\_2/TCP\_fork\_server\_client$ cat test\_file.txt   
Error: Requested File does not exist on the server.  
kanti@kanti-linux:~/kanti\_sem\_3/207/Assgn\_2/TCP\_fork\_server\_client$

**3. Concurrent Multiprocessing (preforking) TCP server and TCP client**

**Server output:**

kanti@kanti-linux:~/kanti\_sem\_3/207/Assgn\_2/TCP\_prefork\_server\_client$ ./server 10000 5  
  
Concurrent preforking TCP ECHO Server Program: listening on socket 10000..  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Number of child processes preforked: 5  
  
Child process:17735 accepted new client connection from 127.0.0.1

Child: 17735 closing the client connection. Returning to the process pool...

**Client Output:**

kanti@kanti-linux:~/kanti\_sem\_3/207/Assgn\_2/TCP\_prefork\_server\_client$ ./client 127.0.0.1 10000  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
TCP test client: This client connects to the concurrent preforked TCP server  
receives number of preforked child processes at the server and prints it.  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
Initiating a connection to TCP server ===> Connection established.  
Number of preforked processes at is server is 5

**4. Concurrent multiprocessing UDP File Server Program (on demand forking)**

**Server output:**

kanti@kanti-linux:~/Assgn\_2/UDP\_fork\_server\_client$ ./server 10000  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Concurrent UDP File Server Program: listening on socket 10000..  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Child processes forked on demand.One per client connection:   
  
forking new child process 10478 to handle the client 127.0.0.1:18376   
File request from client: one\_k\_file.txt   
File name matched: one\_k\_file.txt  
10478: File transfer done.Child Exiting..  
forking new child process 10489 to handle the client 127.0.0.1:4259   
File request from client: t\_k\_file.txt   
t\_k\_file.txt ==> File does not exist on the server.   
10489: Error msg sent. child exiting...  
forking new child process 10496 to handle the client 127.0.0.1:35546   
File request from client: one\_k\_file.txt   
File name matched: one\_k\_file.txt  
10496: File transfer done.Child Exiting..

**Client Output:**

kanti@kanti-linux:~/Assgn\_2/UDP\_fork\_server\_client$ ./client 127.0.0.1 10000 one\_k\_file.txt   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
UDP test client program. Requesting one\_k\_file.txt file from server.  
File will be stored in the client local current directory as "test\_file.txt"  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Sending datagram requesting file transfer..  
File transfer successful. Closing the socket.Exiting the child.  
kanti@kanti-linux:~/kanti\_sem\_3/207/Assgn\_2/UDP\_fork\_server\_client$ ./client 127.0.0.1 10000 t\_k\_file.txt  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
UDP test client program. Requesting t\_k\_file.txt file from server.  
File will be stored in the client local current directory as "test\_file.txt"  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Sending datagram requesting file transfer..  
 Error: Requested File does not exist on the server.  
  
kanti@kanti-linux:~/kanti\_sem\_3/207/Assgn\_2/UDP\_fork\_server\_client$ cat test\_file.txt   
kanti@kanti-linux:~/kanti\_sem\_3/207/Assgn\_2/UDP\_fork\_server\_client$ ./client 127.0.0.1 10000 one\_k\_file.txt   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
UDP test client program. Requesting one\_k\_file.txt file from server.  
File will be stored in the client local current directory as "test\_file.txt"  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Sending datagram requesting file transfer..  
File transfer successful. Closing the socket.Exiting the child.  
kanti@kanti-linux:~/kanti\_sem\_3/207/Assgn\_2/UDP\_fork\_server\_client$ cat test\_file.txt   
start test\_file\_1  
Given a pathname for a file, open() returns a file descriptor, a small, non-negative integer for use in subsequent system calls (read(2), write(2), lseek(2), fcntl(2), etc.). The file descriptor returned by a successful call will be the lowest-numbered file descriptor not currently open for the process.  
The new file descriptor is set to remain open across an execve(2) (i.e., the FD\_CLOEXEC file descriptor flag described in fcntl(2) is initially disabled). The file offset is set to the beginning of the file (see lseek(2)).  
  
A call to open() creates a new open file description, an entry in the system-wide table of open files. This entry records the file offset and the file status flags (modifiable via the fcntl() F\_SETFL operation). A file descriptor is a reference to one of these entries; this reference is unaffected if pathname is subsequently removed or modified to refer to a different file. The new open file description is initially not shared with any other process, but sharing may arise via fork(2).  
end test\_file\_1  
kanti@kanti-linux:~/kanti\_sem\_3/207/Assgn\_2/UDP\_fork\_server\_client$

**5. Concurrent multiprocessing preforking UDP Server Program:**

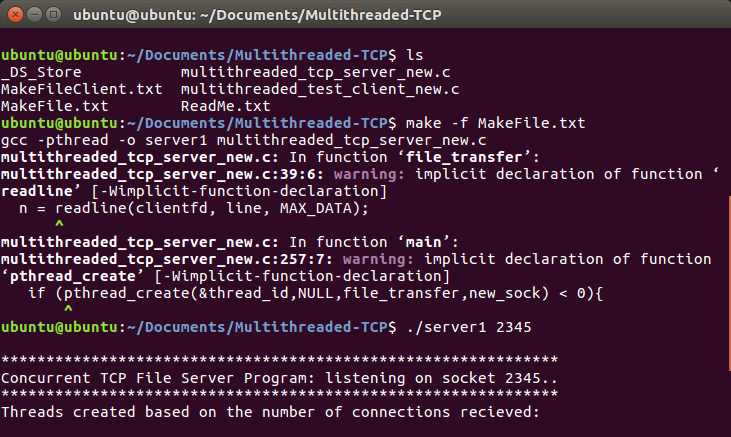
**Server output:**

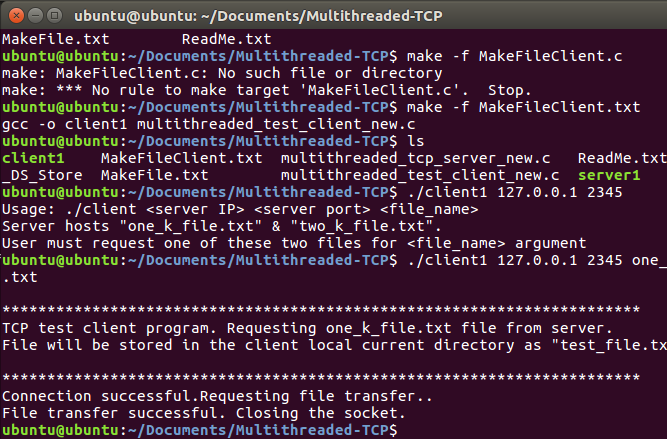
kanti@kanti-linux:~/kanti\_sem\_3/207/Assgn\_2/UDP\_prefork\_server\_client$ ./server 10000 7  
  
Concurrent preforking UDP Server Program   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Number of child processes preforked: 7  
  
Child: 4753 forked....  
Child: 4755 forked....  
Child: 4756 forked....  
Child: 4754 forked....  
Child: 4757 forked....  
Child: 4758 forked....  
Child: 4759 forked....  
Child 4753: Request received from client: 127.0.0.1:54453  
Child 4753: response sent to 127.0.0.1:54453  
^C  
 **Client Output:**

kanti@kanti-linux:~/Assn\_2/UDP\_prefork\_server\_client$ ./client 127.0.0.1 10000  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
UDP test client: Connects to the concurrent preforked UDP server  
receives number of preforked child processes at the server and prints it.  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
Request sent......waiting for server's response  
  
Child pid 4753: Number of preforked processes at is server is 7  
kanti@kanti-linux:~/Assgn\_2/UDP\_prefork\_server\_client$

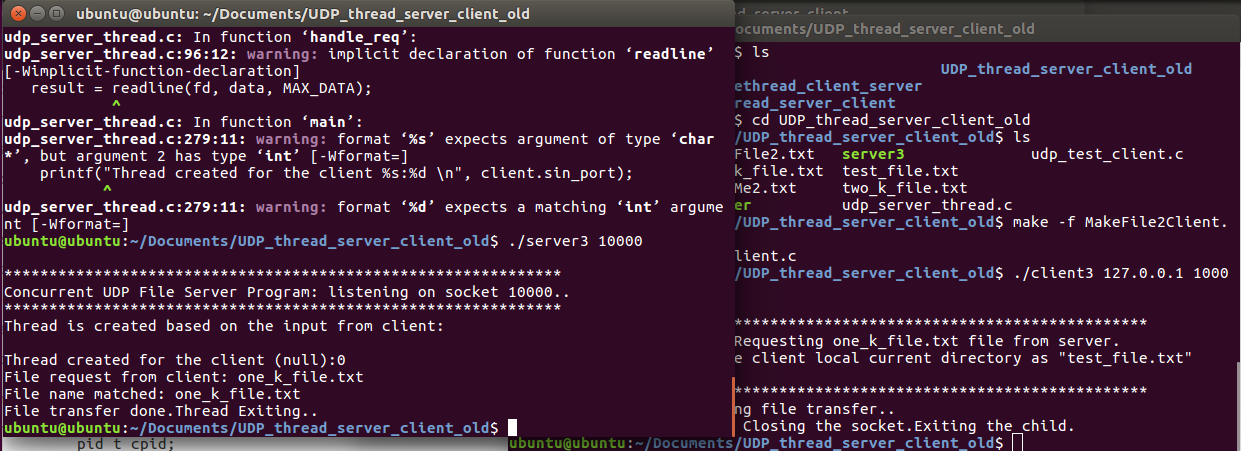
**6. Concurrent Multithreading server and client with one thread per request**

**TCP:**

****

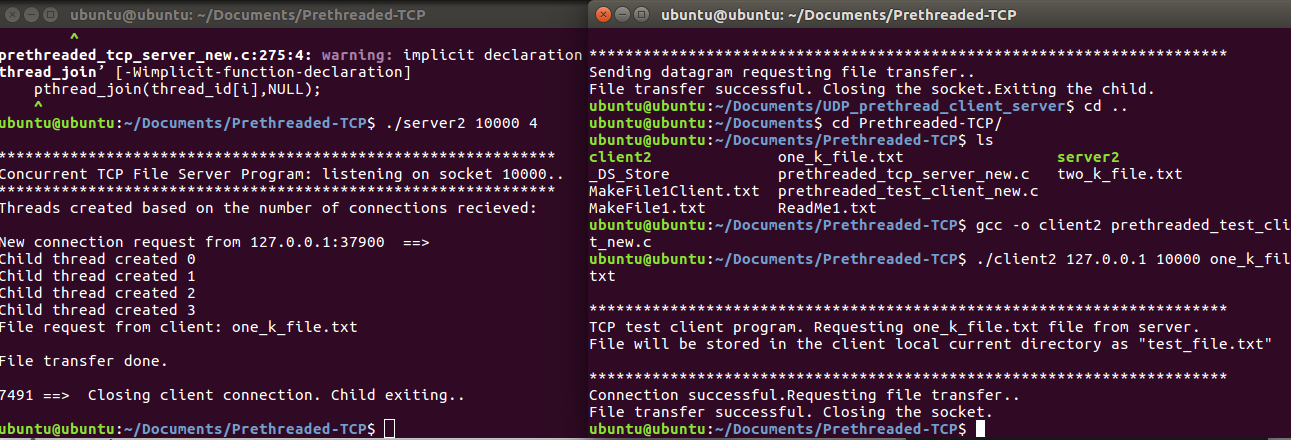
****

**UDP:**

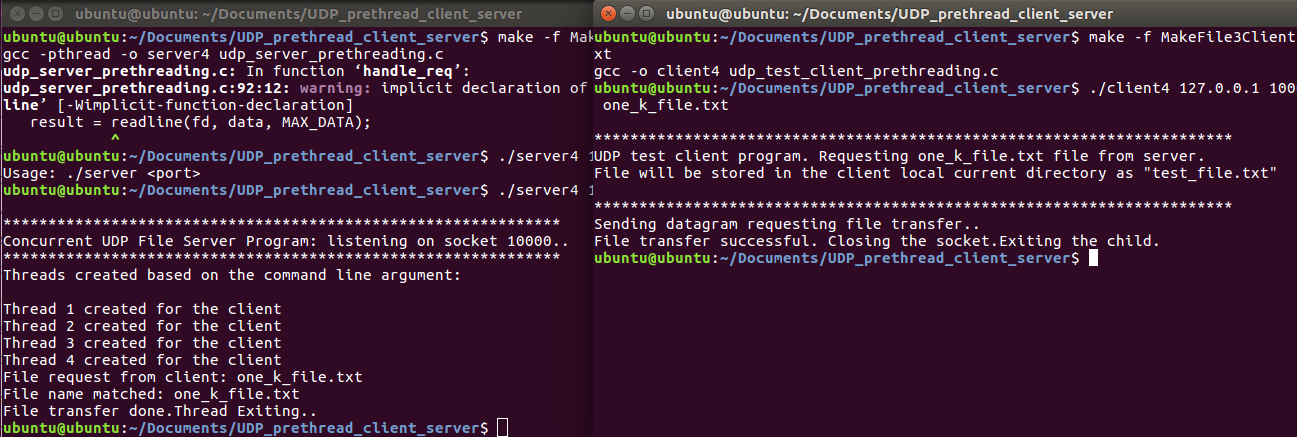
****

**5. Concurrent Pre-threaded Multithreading server**

**TCP:**

****

**UDP:**

****